Partition into heapable sequences, heap tableaux and a multiset extension of Hammersley's process

Gabriel Istrate, Cosmin Bonchiş*

Abstract

We investigate partitioning of integer sequences into heapable subsequences (previously defined and established by Mitzenmacher et al.
[BHMZ11]). We show that an extension of patience sorting computes the decomposition into a minimal number of heapable subsequences (MHS). We connect this parameter to an interactive particle system, a multiset extension of Hammersley's process, and investigate its expected value on a random permutation. In contrast with the (well studied) case of the longest increasing subsequence, we bring experimental evidence that the correct asymptotic scaling is $\frac{1+\sqrt{5}}{2} \cdot \ln(n)$. Finally we give a heap-based extension of Young tableaux, prove a hook inequality and an extension of the Robinson-Schensted correspondence.

1 Introduction

Patience sorting [Mal63] and *the longest increasing (LIS) sequence* are well-studied topics in combinatorics. The analysis of the expected length of the LIS of a random permutation is a classical problem displaying interesting connections with the theory of interacting particle systems [AD99]

*Dept. of Computer Science, West University of Timişoara, Timişoara, Romania. and e-Austria Research Institute, Bd. V. Pârvan 4, cam. 045 B, Timişoara, RO-300223, Romania. Corresponding author's email: gabrielistrate@acm.org

and that of combinatorial Hopf algebras [Hiv07]. Recursive versions of patience sorting are involved (under the name of *Schensted procedure* [Sch61]) in the theory of Young tableaux. A wonderful recent reference for the rich theory of the longest increasing sequences (and substantially more) is [Rom14].

Recently Mitzenmacher et al. [BHMZ11] introduced, under the name of *heapable sequence*, an interesting variation on the concept of increasing sequences. Informally, a sequence of integers is heapable if it can be successively inserted into a (not necessarily complete) binary tree satisfying the heap property without having to resort to node rearrangements. Mitzenmacher et al. showed that the longest heapable subsequence in a random permutation grows linearly (rather than asymptotically equal to $2\sqrt{n}$ as does LIS) and raised as an open question the issue of extending the rich theory of LIS to the case of heapable sequences.

In this paper we partly answer this open question: we define a family $MHS_k(X)$ of measures (based on decomposing the sequence into subsequences heapable into a min-heap of arity at most k) and show that a variant of patience sorting correctly computes the values of these parameters. We show that this family of measures forms an infinite hierarchy, and investigate the expected value of parameter $MHS_2[\pi]$, where π is a random permutation of order n. Unlike the case $k = 1$ where $E[MHS_1[\pi]] = E[LDS[\pi]] \sim 2\sqrt{n}$, we argue that in the case $k \geq 2$ the correct scaling is logarithmic, bringing experimental evidence that the precise scaling is $E[MHS_2[\pi]] \sim \phi \ln n$, where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. The analysis exploits the connection with a new, multiset extension of the Hammersley-Aldous-Diaconis process [AD95], an extension that may be of independent interest. Finally, we introduce a heap-based generalization of Young tableaux. We prove (Theorem 6 below) a hook inequality related to the hook formula for Young tableaux [FRT54] and Knuth's hook formula for heap-ordered trees [Knu98], and (Theorem 8) an extension of the Robinson-Schensted (R-S) correspondence.

2 Preliminaries

For $k \geq 1$ define alphabet $\Sigma_k = \{1, 2, \ldots, k\}$. Define as well $\Sigma_\infty = \cup_{k\geq 1}\Sigma_k$. Given words x, y over Σ_∞ we will denote by $x \sqsubseteq y$ the fact that x is a prefix of y. The set of (non-strict) prefixes of x will be denoted by $Pref(x)$. Given

words $x, y \in \Sigma_\infty^*$ define the *prefix partial order* $x \preceq_{ppo} y$ as follows: If $x \sqsubseteq y$ then $x \preceq_{ppo} y$. If $x = za$, $y = zb$, $a, b \in \Sigma_\infty$ and $a < b$ then $x \preceq_{ppo} y$. $\preceq_{ppo}$ is the transitive closure of these two constraints. Similarly, the *lexicographic partial order* $\preceq_{lex}$ is defined as follows: If $x \sqsubset y$ then $x \preceq_{lex} y$. If $x = za$, $y = zb$, $a, b \in \Sigma_\infty$ and $a < b$ then $x \preceq_{lex} y$. $\preceq_{lex}$ is the transitive closure of these two constraints.

A *k-ary tree* is a finite, $\preceq_{ppo}$-closed set T of words over alphabet $\Sigma_k = \{1, 2, \ldots, k\}$. That is, we impose the condition that positions on the same level in a tree are filled preferentially from left to right. The *position $pos(x)$ of node x in a k-ary tree* is the string over alphabet $\{1, 2, \ldots, k\}$ encoding the path from the root to the node (e.g. the root has position λ, its children have positions $1, 2, \ldots, k$, and so on). A *k-ary (min)-heap* is a function $f : T \to \mathbf{N}$ monotone with respect to pos, i.e. $(\forall x, y \in T), [pos(x) \sqsubseteq pos(y)] \Rightarrow [f(x) \leq f(y)]$.

A *(binary min-)heap* is a binary tree, not necessarily complete, such that $A[parent[x]] \leq A[x]$ for every non-root node x. If instead of binary we require the tree to be k-ary we get the concept of k-ary min-heap.

A sequence $X = X_0, \ldots, X_{n-1}$ is *k-heapable* if there exists some k-ary tree T whose nodes are labeled with (exactly one of) the elements of X, such that for every non-root node X_i and parent X_j, $X_j \leq X_i$ and $j < i$. In particular a 2-heapable sequence will simply be called *heapable* [BHMZ11]. Given sequence of integer numbers X, denote by $MHS_k(X)$ *the smallest number of heapable (not necessarily contiguous) subsequences* one can decompose X into. $MHS_1(X)$ is equal [LP94] to the *shuffled up-sequences (SUS)* measure in the theory of presortedness.

Example 1. *Let $X = [2, 4, 3, 1]$. Via patience sorting $MHS_1(X) = SUS(X) = 3$. $MHS_2(X) = 2$, since subsequnces $[2, 4, 3]$ and $[1]$ are 2-heapable. On the other hand, for every $k \geq 1$, $MHS_k([k, k-1, \ldots, 1]) = k$.*

Analyzing the behavior of LIS relies on the correspondence between longest increasing sequences and an interactive particle system [AD95] called the *Hammersley-Aldous-Diaconis (shortly, Hammersley or HAD) process*. We give it the multiset generalization displayed in Figure 1. Technically, to recover the usual definition of Hammersley's process one should take $X_a > X_{t+1}$ (rather than $X_a < X_{t+1}$). This small difference arises since we want to capture $MHS_k(\pi)$, which generalizes $LDS(\pi)$, rather than $LIS(\pi)$ (captured by Hammersley's process). This slight difference is, of course, inconsequential: our definition is simply the "flipped around

- A number of individuals appear (at integer times $i \geq 1$) as random numbers X_i, uniformly distributed in the interval $[0, 1]$.
- Each individual is initially endowed with k "lifelines".
- The appearance of a new individual X_{t+1} subtracts a life from the largest individual $X_a < X_{t+1}$ (if any) still alive at moment t.

Figure 1: HAD_k, the multiset Hammersley process with k lifelines.

the midpoint of segment [0,1]" version of such a generalization, and has similar behaviour).

3 A greedy approach to computing MHS_k

First we show that one can combine patience sorting and the greedy approach in [BHMZ11] to obtain an algorithm for computing $MHS_k(X)$. To do so, we must adapt to our purposes some notation in that paper.

A binary tree with n nodes has $n + 1$ positions (that will be called *slots*) where one can add a new number. We will identify a slot with the minimal value of a number that can be added to that location. For heap-ordered trees it is the value of the parent node. Slots easily generalize to forests. The number of slots of a forest with d trees and n nodes is $n + d$.

Given a binary heap forest T, the *signature of* T denoted $sig(T)$, is the vector of the (values of) free slots in T, in sorted (non-decreasing) order. Given two binary heap forests T_1, T_2, T_1 *dominates* T_2 if $|sig_{T_1}| \leq |sig_{T_2}|$ and inequality $sig_{T_1}[i] \leq sig_{T_2}[i]$ holds for all $1 \leq i \leq |sig_{T_1}|$.

Theorem 1. *For every fixed $k \geq 1$ there is a polynomial time algorithm that, given sequence $X = (X_0, \dots, X_{n-1})$ as input, computes $MHS_k(X)$.*

Proof. We use the greedy approach of Algorithm 3.1. Proving correctness of the algorithm employs the following

Algorithm 3.1: GREEDY(W)

INPUT $W = (w_1, w_2, \ldots, w_n)$ a list of integers.
Start with empty heap forest $T = \emptyset$.
for i in range(n):
    if (there exists a slot where X_i can be inserted):
        insert X_i in the slot with the lowest value
  else :
        start a new heap consisting of X_i only.

Lemma 1. *Let T_1, T_2 be two heap forests such that T_1 dominates T_2. Insert a new element x in both T_1 and T_2: greedily in T_1 (i.e. at the largest slot with value less or equal to x, or as the root of a new tree, if no such slot exists) and arbitrarily in T_2, obtaining forests T_1', T_2', respectively. Then T_1' dominates T_2'.*

Proof. First note that, by domination, if no slot of T_1 can accomodate x (which, thus, starts a new tree) then a similar property is true in T_2 (and thus x starts a new tree in T_2 as well).

Let $sig_{T_1} = (a_1, a_2, \ldots)$ and $sig_{T_2} = (b_1, b_2, \ldots)$ be the two signatures. The process of inserting x can be described as adding two copies of x to the signature of $T_1(T_2)$ and (perhaps) removing a label $\leq x$ from the two signatures. The removed label is a_i, the largest label $\leq x$, in the case of greedy insertion into T_1. Let b_j be the largest value (or possibly none) in T_2 less or equal to x. Some b_k less or equal to b_j is replaced by two copies of x in T_2. The following are true:

- The length of $sig_{T_1'}$ is at most that of $sig_{T_2'}$.
- The element b_k (if any) deleted by x from T_2 satisfies $b_k \leq x$. Its index in T_2 is less or equal to i.
- The two x's are inserted to the left of the deleted (if any) positions in both T_1 and T_2.

Consider some position l in $sig_{T_1'}$. Our goal is to show that $a_l' \leq b_l'$. Several cases are possible:

- $l < k$. Then $a_l' = a_l$ and $b_l' = b_l$.
- $k \leq l < j$. Then $a_l' = a_l$ and $b_l' = b_{l+1} \geq a_l$.

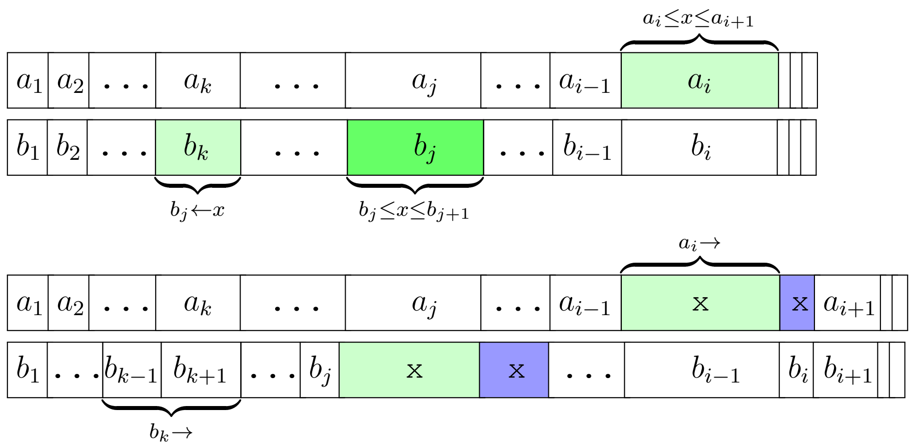


Figure 2: The argument of Lemma 1. Pictured vectors (both initial and resulting) have equal lengths (which may not always be the case).

- $j \leq l \leq i+k-1$. Then $a'_l \leq x$ and $b'_l \geq x$.
- $l > i+k-1$. Then $a'_l = a_{l-k+1}$ and $b'_l = b_{l-k+1}$.

□

□

Let X be a sequence of integers, OPT be an optimal partition of X into k-heapable sequences and Γ be the solution produced by GREEDY. Applying Lemma 1 repeatedly we infer that whenever GREEDY adds a new heap the same thing happens in OPT. Thus the number of heaps created by Greedy is optimal, which means that the algorithm computes $MHS_k(X)$. □ □

Trivially $MHS_k(X) \leq MHS_{k-1}(X)$. On the other hand

Theorem 2. *The following statements (proved in the Appendix) are true for every $k \geq 2$: (a). there exists a sequence X such that $MHS_k(X) < MHS_{k-1}(X) < \ldots < MHS_1(X)$; (b). $\sup_X [MHS_{k-1}(X) - MHS_k(X)] = \infty$.*

4 The connection with the multiset Hammersley process

Denote by $MinHAD_k(n)$ the random variable denoting *the number of times i in the evolution of process HAD_k up to time n when the newly inserted particle X_i has lower value than all the existing particles at time i.* The observation from [Ham72, AD95] generalizes to:

Theorem 3. *For every fixed $k, n \geq 1$ $E_{\pi \in S_n}[MHS_k(\pi)] = E[MinHAD_k(n)]$.*

Proof Sketch: W.h.p. all X_i's are different. We will thus ignore in the sequel the opposite alternative. Informally minima correspond to new heaps and live particles to slots in these heaps (cf. also Lemma 1). □

5 The asymptotic behavior of $E[MHS_2[\pi]]$

The asymptotic behavior of $E[MHS_1[\pi]]$ where π is a random permutation in S_n is a classical problem in probability theory: results in [Ham72], [LS77], [VK77], [AD95] show that it is asymptotically equal to $2\sqrt{n}$.

A simple lower bound valid for all values of $k \geq 1$ is

Theorem 4. *For every fixed $k, n \geq 1$*

$$E_{\pi \in S_n}[MHS_k(\pi)] \geq H_n, \text{ the } n\text{'th harmonic number.} \tag{1}$$

Proof. For $\pi \in S_n$ the set of its *minima* is defined as $Min(\pi) = \{j \in [n] : \pi[j] < \pi[i] \text{ for all } 1 \leq i < j\}$ (and similarly for maxima). It is easy to see that $MHS_k[\pi] \geq |Min[\pi]|$. Indeed, every minimum of π must determine the starting of a new heap, no matter what k is. Now we use the well-known formula $E_{\pi \in S_n}[|Min[\pi]|] = E_{\pi \in S_n}[|Max[\pi]|] = H_n$ [Knu98]. □

□

To gain insight in the behavior of process HAD_2 we note that, rather than giving the precise values of $X_0, X_1, \ldots, X_t \in [0,1]$, an equivalent random model inserts X_t uniformly at random in any of the $t+1$ possible positions determined by $X_0, X_1, \ldots, X_{t-1}$. This model translates into the following equivalent combinatorial description of HAD_k: word w_t over the alphabet $\{-1, 0, 1, 2\}$ describes the state of the process at time t. Each

w_t conventionally starts with a -1 and continues with a sequence of 0, 1's and 2's, informally the "number of lifelines" of particles at time t. For instance $w_0 = 0$, $w_1 = 02$, w_2 is either 022 or 012, depending on $X_0 <> X_1$, and so on. At each time t a random letter of w_t is chosen (corresponding to a position for X_t) and we apply one of the following transformations, the appropriate one for the chosen position:

- *Replacing -10^r by -10^r2:* This is the case when X_t is the smallest particle still alive, and to its right there are $r \geq 0$ dead particles.
- *Replacing 10^r by $0^{r+1}2$:* Suppose that X_a is the largest live label less or equal to X_t, that the corresponding particle X_a has one lifetime at time t, and that there are r dead particles between X_a and X_t. Adding X_t (with multiplicity two) decreases multiplicity of X_a to 0.
- *Replacing 20^r by 10^r2:* Suppose that X_a is the largest label less or equal to X_t, its multiplicity is two, and there are $r \geq 0$ dead particles between X_a and X_t. Adding X_t removes one lifeline from particle X_a.

Simulating the (combinatorial version of the) Hammersley process with two lifelines confirms the fact that $E[MHS_2(\pi)]$ grows significantly slower than $E[MHS_1(\pi)]$: The x-axis in the figure is logarithmic. The scaling is clearly different, and is consistent (see inset) with logarithmic growth (displayed as a straight line on a plot with log-scaling on the x-axis). Experimental results (see the inset/caption of Fig. 3) suggest the following bold

Conjecture 1. *We have $\lim_{n\to\infty} \frac{E[MHS_2[\pi]]}{\ln(n)} = \phi$, with $\phi = \frac{1+\sqrt{5}}{2}$ the golden ratio. More generally, for an arbitrary $k \geq 2$ the relevant scaling is*

$$\lim_{n\to\infty} \frac{E[MHS_2[\pi]]}{\ln(n)} = \frac{1}{\phi_k}, \tag{2}$$

where ϕ_k is the unique root in $(0,1)$ of equation $X^k - X^{k-1} + \ldots + X = 1$.

We plan to present the experimental evidence for the truth of equation (2) and a nonrigorous, "physics-like" justification, together with further insights on the so-called *hydrodynamic behavior* [Gro02] of the HAD_k process in subsequent work [IB15]. For now we limit ourselves to showing that one can (rigorously) perform a first step in the analysis of the HAD_2 process: we prove convergence of (some of) its structural characteristics. This will likely be useful in a full rigorous proof of Conjecture 1.

Denote by L_t the number of digits 1+2, and by C_t the number of ones in w_t. Let $l(t) = E[\frac{L(t)}{t}]$, $c(t) = E[\frac{C(t)}{t}]$. $l(t), c(t)$ always belong to $[0,1]$.

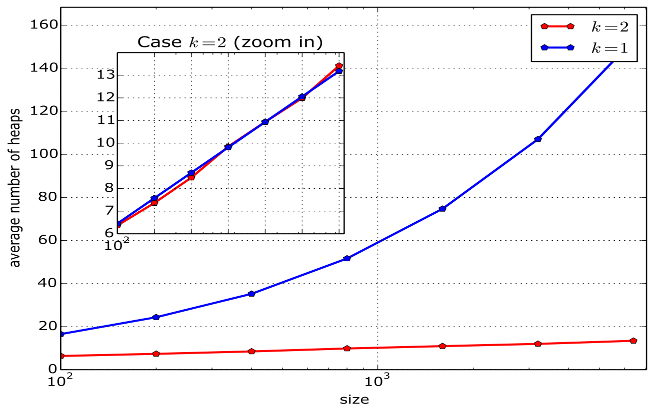


Figure 3: Scaling of expected value of $MHS_k[\pi]$ for $k = 1, 2$. The inset shows $E[MHS_2[\pi]]$ (red) versus $\phi \cdot \ln(n) + 1$ (blue). The fit is strikingly accurate.

Theorem 5. *There exist constants $l, c \in [0, 1]$ such that $l(t) \to l$, $c(t) \to c$.*

Proof Sketch: We use a standard tool, *subadditivity*: if sequence a_n satisfies $a_{m+n} \leq a_m + a_n$ for all $m, n \geq 1$ then (by Fekete's Lemma ([Ste97] pp. 3, [Szp01]) $\lim_{n\to\infty} a_n/n$ exists. We show in the Appendix that this is the case for two independent linear combinations of $l(t)$ and $c(t)$. $\square$

Experimentally (and nonrigorously) $l = \phi - 1 = \frac{\sqrt{5}-1}{2}$ and $c = \frac{3-\sqrt{5}}{2}$. "Physics-like" nonrigorous arguments then imply the desired scaling. An additional ingredient is that digits 0/1/2 are uniformly distributed (conditional on their density) in a large w_t. This is intuitively true since for large t the behavior of the HAD_k process is described by a compound Poisson process. We defer more complete explanations to [IB15].

6 Heap tableaux, a hook inequality and a generalization of the Robinson-Schensted Correspondence.

Finally, we present an extension of Young diagrams to heap-based tableaux. All proofs are given in the Appendix. A *(k-)heap tableau* T is

k-ary min-heap of integer vectors, so that for every $r \in \Sigma_k^*$, the vector V_r at address r is nondecreasing. We formally represent the tableau as a function $T : \Sigma_k^* \times \mathbf{N} \rightarrow \mathbf{N} \cup \{\perp\}$ such that (a). T has *finite support:* the set $dom(T) = \{(r,a) : T(r,a) \neq \perp\}$ of nonempty positions is finite. (b). T is $\sqsubseteq$*-nondecreasing*: if $T(r,a) \neq \perp$ and $q \sqsubset r$ then $T(q,a) \neq \perp$ and $T(q,a) \leq T(r,a)$. In other words, $T(\cdot, a)$ is a min-heap. (c). T is *columnwise increasing*: if $T(r,a) \neq \perp$ and $b < a$ then $T(r,b) \neq \perp$ and $T(r,b) < T(r,a)$. That is, each column V_r is increasing. The *shape of* T is the heap $S(T)$ where node with address r holds value $|V_r|$.

A tableau is *standard* if (e). for all $1 \leq i \leq n = |dom(T)|$, $|T^{-1}(i)| = 1$ and (f). If $x \leq_{lex} y$ and $T(y,1) \neq \perp$ then $\perp \neq T(x,1) \leq T(y,1)$. I.e., labels in the first heap H_1 are increasing from left to right and top to bottom.

Example 2. *A heap tableau T_1 with 9 elements is presented in Fig. 4 (a) and as a Young-like diagram in Fig. 4 (b). Note that: (i). Columns correspond to rows of T_1 (ii). Their labels are in Σ_2^*, rather than* $\mathbf{N}$. *(iii). Cells may contain $\perp$. (iv). Rows need not be increasing, only min-heap ordered.*

One important drawback of our notion of heap tableaux above is that they do not reflect the evolution of the process HAD_k the way ordinary Young tableaux do (on their first line) for process HAD_1 via the Schensted procedure [Sch61]: A generalization with this feature would seem to require that each cell contains not an integer but a *multiset* of integers. Obtaining such a notion of tableau is part of ongoing research.

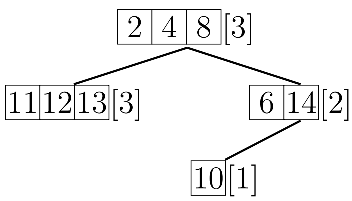
However, we can motivate our definition of heap tableau by the first application below, a hook inequality for such tableaux. To explain it, note that heap tableaux generalize both heap-ordered trees and Young tableaux. In both cases there exist hook formulas that count the number of ways to fill in a structure with n cells by numbers from 1 to n: [FRT54] for Young tableaux and [Knu98] (Sec.5.1.4, Ex.20) for heap-ordered trees. It is natural to wonder whether there exists a hook formula for heap tableaux that provides a common generalization of both these results.

Theorem 6 gives a partial answer: not a formula but a *lower bound.* To state it, given $(\alpha, i) \in dom(T)$, define the *hook length* $H_{\alpha,i}$ to be the cardinal of set $\{(\beta, j) \in dom(T) : [(j = i) \wedge (\alpha \sqsubseteq \beta)] \vee [(j \geq i) \wedge (\alpha = \beta)]\}$. For example, Fig. 4(c). displays the hook lengths of cells in T_1.

Theorem 6. *Given $k \geq 2$ and a k-shape S with n free cells, the number of ways to create a heap tableau T with shape S by filling its cells with numbers* $\{1, 2, \ldots, n\}$

is at least $\frac{n!}{\prod_{(\alpha,i)\in dom(T)} H_{\alpha,i}}$. *The bound is tight for Young tableaux [FRT54], heap-ordered trees [Knu98], and infinitely many other examples, but is also* **not** *tight for infinitely many (counter)examples.*

We leave open the issue whether one can tighten up the lower bound above to a formula by modifying the definition of the hook length $H_{\alpha,i}$.



	λ	0	1	00	01	10
1	2	11	6	$\perp$	$\perp$	10
2	4	12	14			
3	8	13				

	λ	0	1	00	01	10
1	6	3	3	$\perp$	$\perp$	1
2	4	2	1			
3	2	1				

Figure 4: (a). Heap tableau T_1 and its shape $S(T_1)$ (in brackets) (b). The equivalent Young tableau-like representation of T_1 and (c). The hook lengths.

We can create k-heap tableaux from integer sequences by a version of the Schensted procedure [Sch61]. Algorithm Schensted-HEAP$_k$ below performs *column insertions* and gives to any bumped element k choices for insertion/bumping, the children of vector V_r, with addresses $r \cdot \Sigma_k$.

Theorem 7. *The result of applying the Schensted-HEAP$_k$ procedure to an arbitrary permutation X is indeed a k-ary heap tableau.*

Example 3. *Suppose we start with T_1 from Fig. 4(a). Then (Fig. 5) 9 is appended to vector V_λ. 7 arrives, bumping 8, which in turn bumps 11. Finally 11 starts a new vector at position 00. Modified cells are grayed.*

Procedure Schensted-HEAP$_k$ does **not** help in computing the longest heapable subsequence: The complexity of computing this parameter is open [BHMZ11], and we make no progress on this issue. On the other hand, we can give a $k \geq 2$ version of the R-S correspondence:

Theorem 8. *For every $k \geq 2$ there exists a bijection between permutations $\pi \in S_n$ and pairs (P, Q) of k-heap tableaux with n elements and identical shape, where Q is a standard tableau.*

Condition "Q is standard" is specific to case $k \geq 2$: heaps simply have "too many degrees of freedom" between siblings. Schensted-HEAP$_k$ solves this problem by starting new vectors from left to right and top to bottom.

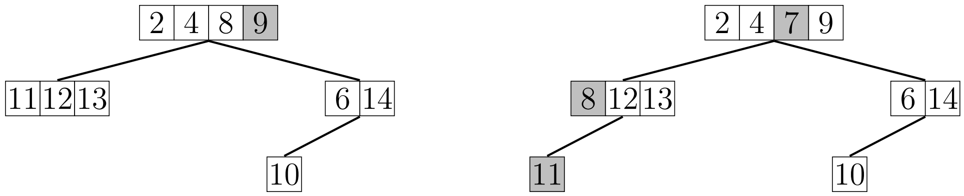


Figure 5: Inserting 9 and 7 into T_1.

Algorithm 6.1: SCHENSTED-HEAP$_k(X = x_0, \ldots, x_{n-1})$

for i in range(n) : $BUMP(x_i, \lambda)$

PROCEDURE BUMP(x, S) : $\quad$ *#S is a set of adresses.*
- Attempt to append x to some $V_r, r \in S$ (perhaps creating it)
  (choose the first r where appending x keeps V_r increasing).

if (this is not possible for **any** vector $V_r, r \in S$) :
- Let B_x be the set of elements of value $> x$,
  in all vectors V_r, $r \in S$ (clearly $B_x \neq \emptyset$)
- Let $y = min\{B_x\}$ and r the address of its vector.
- Replace y by x into V_r
- $BUMP(y, r \cdot \Sigma_k)$ $\quad$ *#bump y into some child of r*

7 Conclusion and Acknowledgments

Our paper raises a large number of open issues. We briefly list a few: Rigorously justify Conjecture 1. Study process HAD_k and its variants [Mon97, CG05]. Reconnect the theory to the analysis of *secretary problems* [AM09, BKK+09]. Find the distribution of $MHS_k[\pi]$. Obtain a hook formula. Define a version of Young tableaux related to process HAD_k.

We plan to address some of these in subsequent work. The most important open problem, however, is *the complexity of computing LHS.*

This research has been supported by CNCS IDEI Grant PN-II-ID-PCE-2011-3-0981 "Structure and computational difficulty in combinatorial optimization: an interdisciplinary approach".

References

[AD95] D. Aldous and P. Diaconis. Hammersley's interacting particle process and longest increasing subsequences. *Probability theory and related fields*, 103(2):199–213, 1995.

[AD99] D. Aldous and P. Diaconis. Longest increasing subsequences: from patience sorting to the Baik-Deift-Johansson theorem. *Bulletin of the American Mathematical Society*, 36(4):413–432, 1999.

[AM09] M. Archibald and C. Martínez. The hiring problem and permutations. *DMTCS Proceedings*, (01):63–76, 2009.

[BHMZ11] J. Byers, B. Heeringa, M. Mitzenmacher, and G. Zervas. Heapable sequences and subseqeuences. In *Proceedings of ANALCO*, pages 33–44, 2011.

[BKK+09] A. Broder, A. Kirsch, R. Kumar, M. Mitzenmacher, E. Upfal, and S. Vassilvitskii. The hiring problem and Lake Wobegon strategies. *SIAM Journal on Computing*, 39(4):1233–1255, 2009.

[CG05] E. Cator and P. Groeneboom. *Hammersley's process with sources and sinks.* Annals of Probability (2005): 879-903.

[FRT54] J.S. Frame and G. Robinson, and R.M. Thrall. *The hook graphs of the symmetric group.* Canad. J. Math 6.316 (1954): C324.

[GNW79] C. Greene and A. Nijenhuis and H.S.Wilf. *A probabilistic proof of a formula for the number of Young tableaux of a given shape.* Advances in Mathematics 31.1 (1979): 104-109.

[Ham72] J. M. Hammersley. A few seedlings of research. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics*, 1972.

[Hiv07] F. Hivert. An introduction to combinatorial Hopf algebras *Physics and theoretical computer science: from numbers and languages to (quantum) cryptography security*, IOS Press, 2007.

[Gro02] P. Groenenboom. Hydrodynamical models for analyzing longest increasing sequences. *Journal of Computational and Applied Mathematics*, 142 (2002), 83–105.

[IB15] G. Istrate and C. Bonchiş. *Hammersley's process with multiple lifelines, and a conjecture on decomposing permutations into heapable sequences* (manuscript in preparation, 2015)

[Knu98] D. Knuth. The Art of Computer Programming. Volume 3: Sorting and Searching. Addison Wesley, 1998.

[LP94] C. Levcopoulos and O. Petersson. Sorting shuffled monotone sequences. *Information and Computation*, 112(1):37–50, 1994.

[LS77] B.F. Logan and L.A. Shepp. A variational problem for random Young tableaux. *Advances in mathematics*, 26(2):206–222, 1977.

[Mal63] C.L. Mallows. Patience Sorting. *SIAM Review*, 5(4), 375–376, 1963.

[Mon97] L. Cuellár Montoya. *A rapidly mixing stochastic system of finite interacting particles on the circle.* Stochastic processes and their applications 67.1 (1997): 69-99.

[Rom14] D. Romik. *The Surprising Mathematics of Longest Increasing Sequences*. Cambridge University Press, 2014.

[Sch61] C. Schensted. *Longest increasing and decreasing subsequences.* Canad. J. Math 13.2 (1961): 179-191.

[Szp01] W. Szpankowski. *Average Case Analysis of Algorithms on Sequences.* Wiley, 2001.

[Ste97] J.M. Steele. *Probability theory and combinatorial optimization*. SIAM, 1997.

[VK77] A. Vershik and S. Kerov. Asymptotics of Plancherel measure of symmetric group and limit form of Young tables. *Doklady Akademii Nauk SSSR*, 233(6):1024–1027,1977.

Appendix

7.1 Proof of Theorem 2

1. For $k \geq 2$, consider the sequence $X = [1, k+1, k, k-1, \cdots, 2]$.

Lemma 2. *We have*

$$MHS_1(X) = k, MHS_2(X) = k-1, \ldots, MHS_k(X) = 1.$$

Proof. Applying the Greedy algorithm we obtain the following heap decompositions:

- $\mathbf{MHS_1(X) = k}$: $H_1 = [1, k+1]$, $H_2 = [k]$, $H_3 = [k-1]$, $\ldots$, $H_k = [2]$.
- $\mathbf{MHS_2(X) = k-1}$: $H_1 = [1, k+1, k]$, $H_2 = [k-1]$, $H_3 = [k-2]$, $\ldots$, $H_{k-1} = [2]$.

  $\vdots$
- $\mathbf{MHS_i(X) = k-i+1}$: $H_1 = [1, k+1, k, \ldots, k-i+2]$, $H_2 = [k-i+1]$, $H_3 = [k-i]$, $\ldots$, $H_{k-i+1} = [k+2]$.

  $\vdots$
- $\mathbf{MHS_k(X) = 1}$: $H_1 = [1, k+1, k, \cdots, 2]$.

□

□

2. Let $k, n \geq 2$. Define sequence

$$\begin{aligned} X^{(k,n)} &= [1, \\ &\quad (2+(k-1)), k, \ldots, 2, \\ &\quad (3+2(k-1)+(k-1)^2), (k+k^2), \ldots, (2+k), \\ &\quad \vdots \\ &\quad \sum_{i=0}^{n}(n+1-i)(k-1)^i, \ldots, 1+\sum_{i=0}^{n-1}(n-i)(k-1)^i] \end{aligned}$$

in other words $X^{(k,n)} = [1, X_1, X_2, \ldots, X_n]$, where for each $1 \leq t \leq n$ the subsequence X_t is $X_t = [\sum_{i=0}^{t}(t+1-i)(k-1)^i, \ldots, 1+\sum_{i=0}^{t-1}(t-i)(k-1)^i]$. X_t has $(k-1)^t + (k-1)^{t+1} + \ldots + 1 = \frac{(k-1)^{t+1}-1}{k-2}$ many elements.

We can see that this sequence is k-heapable, thus $MHS_k(X) = 1$: $|X_t| = (k-1)|X_{t-1}|+1 < k|X_t|$, and every number in X_t is larger than

every number in X_{t-1}. Thus we can arrange the X_t's on (incomplete) heap levels, with every node in X_t a child of some node in X_{t-1}.

Theorem 9. *We have*

$$MHS_{k-1}(X^{(k,n)}) \quad = \quad n+1.$$

Proof. We apply the GREEDY algorithm. After sequence X_1 two $(k-1)$-heaps are created. H_1 has two full levels, H_2 contains only the root 2. Sequence X_2 has length k^2. $(k-1)^2$ elements go on the third level of H_1. $k-1$ elements go on the second level of X_2. The remaining $k^2-(k-1)^2-2(k-1) = 1$ element starts a new heap H_3.

By induction we easily prove the following

Lemma 3. *For every $t \geq 1$, the $\frac{(k-1)^{t+1}-1}{k-2}$ elements of X_t go via GREEDY as follows:*

- *$(k-1)^t$ of them go on level t of H_1,*
- *$(k-1)^{t-1}$ of them go on level $t-1$ of H_2,*
- *...*
- *$k-1$ of them go on the first level of H_t.*

The remaining $\frac{(k-1)^{t+1}-1}{k-2} - \sum_{i=1}^{t}(k-1)^i = 1$ element starts a new heap H_{t+1}.

□

□

7.2 Proof of Theorem 5

First sequence: Define a_n to be the expected cardinality of the multiset of **slots** (particles lifelines in process HAD_2)) at moment n. Clearly $a_n/n = 2l(n) - c(n)$. Also, given $Z = (Z_0, Z_1, \ldots, Z_{n-1})$ a finite trajectory in [0,1] **and an initial set of slots** T, denote by $s(Z;T)$ the multiset of **particles (slots) added during** Z that are still alive at the end of the trajectory Z, if **at time** $t=0$ **the process started with the slots in** T (omitting the second

argument if $T = \emptyset$), and $a(Z;T) = |s(Z;T)|$. Finally denote by $v(Z;T)$ the submultiset of $s(Z;T)$ consisting of elements with multiplicity two, and by $l(Z;T) = |v(Z;T)|$.

Subadditivity of a_n will follow from the fact that the property holds *on each trajectory*: If $X = (X_0, \ldots, X_{n-1})$ and $Y_m = (X_n \ldots X_{n+m-1})$ then in fact we can show that

$$a(XY_m) \leq a(X) + a(Y_m). \tag{3}$$

Clearly $a_n = E_{|X|=n}[a(X)]$ so (2) implies that a_n is subadditive. It turns out that, together with (3), we will need to simultaneously prove that

$$s(Y_m) \subseteq s(Y_m; s(X)) \text{ (as multisets)} \tag{4}$$

We prove (3) and (4) by induction on $m = |Y_m|$. Clearly the inclusion is true if $m = 0$. Let $Y_m = Y_{m-1}X_{n+m-1}$ and $s(XY_m) = W_m \cup Z_m$, with $W_m = s(X) \cap s(XY_m)$, $Z_m = Y_m \cap s(XY_m)$.

$s(XY_m)$ modifies $s(XY_{m-1})$ by adding two copies of X_{n+m-1} to W_m and, perhaps, erasing some p_m, the largest element (if any) in $s(XY_{m-1})$ smaller or equal to X_{n+m-1}. Thus $a(XY_m) - a(XY_{m-1}) \in \{1, 2\}$.

Similarly, $s(Y_m)$ modifies $s(Y_{m-1})$ by adding two copies of X_{n+m-1} and, perhaps, erasing some r_m, the largest element (if any) **in** $s(Y_{m-1})$ smaller or equal to X_{n+m-1}. Thus $a(Y_m) - a(Y_{m-1}) \in \{1, 2\}$.

All that remains in order to prove that $a(XY_m) - a(Y_m) \leq a(XY_{m-1}) - a(Y_{m-1})$ (and thus establish inequality (2) inductively for m as well) is that $(a(Y_m) - a(Y_{m-1}) = 1) \Rightarrow (a(XY_m) - a(XY_{m-1}) = 1)$. This follows easily from inductive hypothesis (4) for $m-1$: if $a(Y_m) - a(Y_{m-1}) = 1$ then some element in $s(Y_{m-1})$ is less or equal to X_{n+m-1}. The same must be true for $s(Y_{m-1}; s(X))$ and hence for $s(XY_{m-1})$ as well (noting, though, that p_m may well be an element of X). Now we have to show that (4) also remains true: clearly the newly added element, X_{n+m-1}, has multiplicity two in both $s(Y_m)$ and $s(Y_m; s(X))$. Suppose we erase some element r_m from $s(Y_{m-1})$. Then r_m belongs to $s(Y_{m-1}; s(X))$, has multiplicity at least one there, and is *the largest element smaller or equal to* X_{n+m-1} *in* $s(Y_{m-1}; s(X)) \cap s(Y_{m-1})$. Thus, when going from $s(Y_{m-1}; s(X))$ to $s(Y_m; s(X))$ we either erase one copy of p_m or do not erase nothing (perhaps we erased some element in $s(X)$, which is not, however, in $s(Y_{m-1}; s(X))$) Suppose, on the other hand that no element in $s(Y_{m-1})$ is smaller or equal to X_{n+m-1}. There may be such an erased element p_m in $s(Y_{m-1}; s(X))$, but *it certainly did not belong to* $s(Y_{m-1})$. In both cases we infer that relation $s(Y_m) \subseteq s(Y_m; s(X))$ is true.

Second sequence:

The proof is very similar to the first one: Define, in a setting similar to that of the first sequence, $u(X,T)$ to be the cardinality of the submultiset of $s(Z,T)$ of elements with multiplicity two. Define a_n to be the expected number of elements with multiplicity two at stage n. That is, $a_n = E_{|X|=n}[u(X)] = l(n) - c(n)$. We will prove by induction on m that if $X = (X_0, \ldots, X_{n-1})$ and $Y_m = (X_n \ldots X_{n+m-1})$ then

$$u(XY_m) \leq u(X) + u(Y_m). \tag{5}$$

The result is clear for $m = 0$. In the general case, $m \geq 1$, $v(XY_m)$ modifies $v(XY_{m-1})$ by adding X_{n+m-1} and, perhaps, erasing some p_m, the largest element (if any) in $s(XY_{m-1})$ smaller or equal to X_{n+m-1} if this element is in $v(XY_{m-1})$. Thus $u(XY_m) - u(XY_{m-1}) \in \{0, 1\}$. Similarly, $u(Y_m)$ modifies $u(Y_{m-1})$ by adding X_{n+m-1} and, perhaps, erasing some r_m, the largest element (if any) **in** $s(Y_{m-1})$, if this element is smaller or equal to X_{n+m-1}. Thus $u(Y_m) - u(Y_{m-1}) \in \{0, 1\}$.

If $u(XY_{m-1}) \leq u(X) - 1 + u(Y_m)$ then clearly $u(XY_m) - u(Y_m) \leq u(X)$. The only problematic case may be when $u(XY_{m-1}) - u(Y_{m-1}) = u(X)$, $u(XY_m) - u(XY_{m-1}) = 1$, $u(Y_m) - u(Y_{m-1}) = 0$. But this means that r_m exists (and is erased from $v(Y_{m-1})$). Since $s(Y_{m-1}) \subseteq s(Y_{m-1}; s(X))$, r_m must be erased from $s(Y_{m-1}; s(X))$. In other words, the bad case above cannot occur.

7.3 Proof of Theorem 6

We use essentially the classical proof based on the *hook walk* from [GNW79], slightly adapted to our framework: Define for a heap table T with n elements

$$F_T = \frac{n!}{\prod_{(\alpha,i)\in dom(T)} H_{\alpha,i}}$$

and $C(T)$, the set of *corners of* T, to be the set of cells (α, i) of T with $H_{\alpha,i} = 1$. Given $\gamma \in C(T)$ define $T_\gamma = T \setminus \{\gamma\}$. We want to prove that

$$\sum_{\gamma\in C(T)} \frac{F_{T_\gamma}}{F_T} \geq 1. \tag{6}$$

(of course, for $k = 1$ we can actually prove equality in Formula 6 above). This will ensure (by induction upon table size) the truth of our lower bound.

- Choose (uniformly at random) a cell (α_1, i_1) of T.
- let $i = 1$.
- while $((\alpha_i, t_i)$ is not a corner of T):
- Choose (α_{i+1}, t_{i+1}) uniformly at random from $H((\alpha_i, t_i)) \setminus \{(\alpha_i, t_i)\}$.
- Let $i = i + 1$.
- Return corner (α_n, i_n).

Figure 6: The hook walk.

We need some more notation: for $(\alpha, i) \in dom(T)$, denote

$$Heap_{\alpha,i} = \{(\beta, i) \in dom(T) : \alpha \sqsubseteq \beta\} \tag{7}$$

the *heap hook of* (α, i), and by

$$Vec_{\alpha,i} = \{(\alpha, j) \in dom(T) : i \leq j\} \tag{8}$$

its *vector hook* (thus $H_{\alpha,i} = |Heap_{\alpha,i}| + |Vec_{\alpha,i}| - 1$).

By applying formulas for F_T, F_{T_γ} we get

$$\begin{aligned} \frac{F_{T_\gamma}}{F_T} &= \frac{1}{n} \cdot \prod_{\gamma \in Heap_{\beta,j}} \frac{H_{\beta,j}}{H_{\beta,j} - 1} \cdot \prod_{\gamma \in Vec_{\beta,j}} \frac{H_{\beta,j}}{H_{\beta,j} - 1} \\ &= \frac{1}{n} \cdot \prod_{\gamma \in Heap_{\beta,j}} (1 + \frac{1}{H_{\beta,j} - 1}) \cdot \prod_{\gamma \in Vec_{\beta,j}} (1 + \frac{1}{H_{\beta,j} - 1}) \end{aligned} \tag{9}$$

We consider the *hook walk on* T, defined in Figure (6).

Interpret terms from the product in formula (9) as probabilities of paths in the hook walk, ending in corner γ, as follows:

- Choose (α_1, i_1) uniformly at random from T (i.e. with probability 1/n).
- Terms (β, i) in the first product whose contribution is $\frac{1}{H_{\beta,i}-1}$ correspond to cells where the walk makes "hook moves" towards γ.

- Terms (β, i) in the second product whose contribution is $\frac{1}{H_{\beta,i}-1}$ correspond to cells where the walk makes "vector moves" towards γ.

Indeed, consider a path $P : (\alpha, i) := (\alpha_1, i_1) \to (\alpha_2, i_2) \to \ldots \to (\alpha_n, i_n) = \gamma$. Define its *hook projection* to be set $A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ and its vector projection to be the set $B = \{i_1, i_2, \ldots, i_n\}$.

Just as in [GNW79], given set of words $A = \{\alpha_1, \ldots \alpha_m\}$, with $\alpha_1 = \alpha$ and $\alpha_i \sqsubset \alpha_{i+1}$ and set of integers $B = \{i_1, \ldots, i_r\}$ with $i_1 = i$ and $i_l < i_{l+1}$, the probability $p(A, B)$ that the hook walk has the hook(vector) projections $A(B)$ (thus starting at (α_1, i_1)) is

$$P(A,B) \leq \prod_{\beta \in A, \beta \neq \alpha_m} (1 + \frac{1}{H_{\beta,i_r} - 1}) \cdot \prod_{i \in B, i \neq i_r} (1 + \frac{1}{H_{\alpha_m,i} - 1}) \tag{10}$$

Indeed, as in [GNW79]

$$\begin{aligned} P(A,B) &= \frac{1}{H_{\alpha_1,i_1} - 1}[P(A - \{\alpha_1\}, B) + P(A, B - \{i_1\})] \leq \\ &\leq \frac{1}{H_{\alpha_1,i_1} - 1}[(H_{\alpha_1,i_r} - 1) + (H_{\alpha_m,i_1} - 1)] \cdot (RHS) \end{aligned} \tag{11}$$

where (RHS) is the right-hand side product in equation (10), and in the second row we used the inductive hypothesis.

For $k = 1$, in [GNW79] we would use an equality of type $H_{\alpha_1,i_1} - 1 = (H_{\alpha_1,i_r} - 1) + (H_{\alpha_m,i_1} - 1)$. For $k \geq 2$ such an equality is no longer true, and we only have inequality

$$H_{\alpha_1,i_1} - 1 \geq (H_{\alpha_1,i_r} - 1) + (H_{\alpha_m,i_1} - 1) \tag{12}$$

leading to a proof of equation (10).

To justify inequality (12), note that, by property (b) of heap tableaux, since $\alpha_1 \sqsubset \alpha_m$,

$$|Vec(\alpha_m, i_1)| \leq |Vec(\alpha_1, i_1)| \tag{13}$$

On the other hand

$$|Heap(\alpha_1, i_1)| \geq |Heap(\alpha_1, i_r)| + (|Heap(\alpha_m, i_1)| - 1). \tag{14}$$

This is true by monotonicity property (c) of heap tableaux: every path present in the heap H_r rooted at (α_1, i_r) is also present in the heap H_1

rooted at (α_1, i_1). Heap H_r is empty below node $\gamma = (\alpha_m, i_r)$, but H_1 contains the subheap rooted at (α_1, i_r) (of size $|Heap(\alpha_1, i_r)|-1$) *any maybe some other subheaps,* rooted at nodes $w \in H_1$ whose correspondent in H_r has no descendents. Summing up equations (13) and (14) we get our desired inequality (12). Example in Figure 7 shows that inequality (12) can be strict: The hook length of $H_{1,\lambda} - 1 = 7$ but $H_{2,\lambda} - 1 = 2 - 1$ and $H_{1,0} - 1 = 2 - 1$. The reason is that the grayed cells are not counted in the hook of $(1, 0)$, but they belong to the hook of $(1, \lambda)$.

	λ	0	1	$\cdots$	1^2	$\cdots$	1^3	$\cdots$	1^4	$\cdots$	1^5
1	8	2	5	$\perp$	4	$\perp$	3	$\perp$	2	$\perp$	1
2	2	1									

Figure 7: Example showing that inequality (12) is strict.

Finally, adding up suitable inequalities (10) we infer that s_γ, the probability that the walk ends up at γ, equal to

$$s_\gamma = \frac{1}{n} \sum p(A, B)$$

(the sum being over all suitable sets A, B) is less or equal than the expansion (9) of $\frac{F_{T_\gamma}}{F_T}$. Since the sum of probabilities adds up to 1, inequality (6) follows.

Let us now deal with examples/counterexamples.

First we present a set of arbitrarily large heap tableaux, different from both heap-ordered trees and Young tableaux, for which the hook inequality is tight: for $r \geq 2, k \geq 1$ consider heap table $T_{r,k}$ (Fig. 8(a)) to have $n = S_{k,r} + k - 1$ nodes, distributed in a complete k-ary tree H_1 with r levels $0, 1, \ldots r - 1$ and $S_{k,r}$ nodes, and then $k - 1$ one-element heaps $H_2, \ldots, H_k$. We employ notation

$$S_{k,l} = 1 + k + \ldots + k^{l-1} = \frac{k^l - 1}{k - 1}$$

The number of ways to fill up such a heap tableau is $\binom{n-1}{k-1} \cdot N_{k,r}$, where $N_{k,r}$ is the number of ways to fill up a complete k-ary tree with r levels.

$$N_{k,r} = \frac{S_{k,r}!}{\prod_{i=0}^{r-1} (S_{k,r-i})^{k^i}}$$

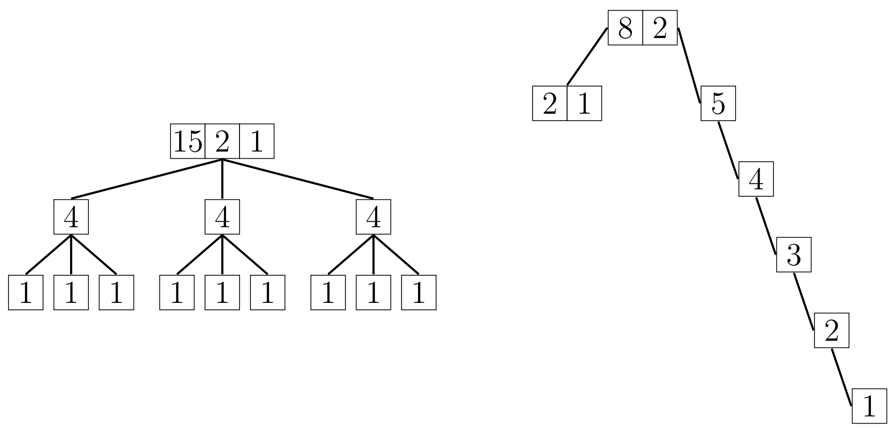


Figure 8: (a). Example $T_{3,3}$. (b). Counterexample W_4. The hook formula is tight for heap tableau (a). but not tight for (b). In both cases cell contents represent the hook lengths.

This happens because for every subset A of $\{2, \ldots, n\}$ of cardinality $k-1$, element 1 together with those not in A can be distributed in H_1 in $N_{k,r}$ ways.

Putting all things together, the total number of fillings of $T_{r,k}$ is

$$\frac{(S_{k,r}+k-2)! \cdot (S_{k,r})!}{(k-1)! \cdot (S_{k,r}-1)! \cdot S_{k,r} \cdot \prod_{i=1}^{r-1}(S_{k,r-i})^{k^i}} = \frac{(n-1)!}{(k-1)! \cdot \prod_{i=1}^{r-1}(S_{k,r-i})^{k^i}}$$

Hook lengths are $1, 2, \ldots, k-1$ (for the nodes in the one-element heaps), $(S_{k,r-i})^{k^i}$ (for the non-root nodes in H_1) and n (for the root node of H_1). The resulting formula

$$\frac{n!}{(k-1)! \cdot n \cdot \prod_{i=1}^{r-1}(S_{k,r-i})^{k^i}} = \frac{(n-1)!}{(k-1)! \cdot \prod_{i=1}^{r-1}(S_{k,r-i})^{k^i}} \tag{15}$$

is the same as the total number computed above.

Now for the counterexamples: consider heap tableaux W_r (Fig. 8(b), identical to the heap tableau in Fig. 7) defined as follows: W_r consists of two heaps, H_1 with cells with addresses $(1,\lambda), (1,0)$, $(1,1), (1,11), \ldots,$ $(1, 1^{2r-3})$, and H_2 with cells with addresses $(2,\lambda), (2,0)$. W_r has $n = 2r+1$ nodes.

Hook values of cells in H_1 are $2r, 2, 2r-3, 2r-4, \ldots, 1$. Hook values of

cells in H_2 are $2, 1$, respectively. Thus the hook formula predicts

$$\frac{(2r+1)!}{2 \cdot 2 \cdot 2r \cdot (2r-3) \cdot (2r-4) \cdot \ldots \cdot 1} = \frac{(2r+1)(2r-1)(2r-2)}{4}$$

ways to fill up the table. If r is even then the number above is **not** an integer, so the hook formula cannot be exact for these tableaux.

7.4 Proof of Theorem 7

We prove that inserting a single integer element x into a heap tableau T results in another heap tableau $T \leftarrow x$. Therefore inserting a permutation X will result in a heap tableau.

By construction, when an element is appended to a vector, the vector remains increasing. Also, if an element y bumps another element z from a vector V (presumed nondecreasing) then z is the smallest such element in V greater than y. Thus, replacing z by y preserves the nondecreasing nature of the vector V.

All we need to verify is that min-heap invariant (b) (initially true for the one-element heap tableau) also remains true when inserting a new element x.

The case when x is appended to V_λ is clear: since invariant (b) was true before inserting x for every address r we have $|V_\lambda| \geq |V_r|$. See the example above when we append $x = 9$. Thus what we are doing, in effect, by appending x to V_λ is start a new heap.

Suppose instead that inserting x bumps element x_1 from V_λ. Necessarily $x < x_1$. Suppose i is the position of x_1 in V_λ, that is x_1 was the root of heap H_i. By reducing the value of the root, the heap H_i still verifies the min-heap invariant. Now suppose x_1 bumps element x_2. We claim that x_2 has rank at most i in its vector. Indeed, the element with rank i in the vector of x_2 was larger than x_1 (by the min-heap property of H_i). So x_2 must have had rank at most i. Let j be this rank.

Since $x_1 < x_2$, by replacing x_2 by x_1 the min-heap property is satisfied "below x_2/x_1". It is satisfied "above x_2" as well, since the parent of x_2 either was (and still is) the root of H_j, a number less or equal to x_1 (in case $j < i$) or is x (in case the rank of x_2 is exactly i).

If x_2 bumps $x_3, \ldots,$ etc we repeat the argument above on the corresponding sub-min-heap tableau.

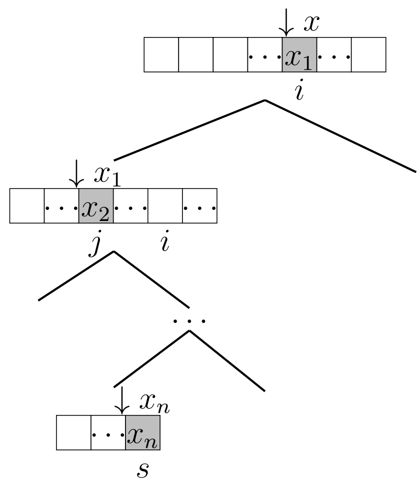


Figure 9: Inserting x and the bumps it determines.

Suppose, finally, that element x_n, bumped from V_α by x_{n-1}, is appended to vector V_β. Let s be the index of x_n in V_β. We claim that $|V_\alpha| \geq s$.

Indeed, x_n is larger than the first $s-1$ elements of $|V_\beta|$. By the min-heap property, it is also larger than the initial $s-1$ elements of $|V_\alpha|$ as well. So its index in V_α before getting bumped could not have been less than s. That means that appending x_n does not violate the min-heap invariant (b).

7.5 Proof of Theorem 8

Given permutation $\sigma \in S_n$, denote by P_σ the heap table obtained by applying the Schensted-HEAP$_k$ algorithm.

Define a second heap table Q_σ as follows: whenever we insert $\sigma(i)$ into P, we record the resulting sequence of bounces and **insert i at the last place involved in the bounces.**

Example 4. *Let* $k = 2$ *and consider the permutation* $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 2 & 6 & 3 & 5 & 1 \end{pmatrix}$. *The two corresponding heap tableaux are constructed below. For drawing convenience, during the insertion process they are not displayed in the heap-like form, but rather in the more compact Young-table equivalent format. The resulting heap-tableaux are displayed in Figure 10.*

	$\leftarrow 4$	$\leftarrow 2$	$\leftarrow 6$	$\leftarrow 3$	$\leftarrow 5$	$\leftarrow 1$
P_σ:	λ 1: 4	λ 0 1: 2 4	λ 0 1: 2 4 2: 6	λ 0 1: 2 4 2: 3 6	λ 0 1: 2 4 2: 3 6 3: 5	λ 0 1 1: 1 4 2 2: 3 6 3: 5
Q_σ:	λ 1: 1	λ 0 1: 1 2	λ 0 1: 1 2 2: 3	λ 0 1: 1 2 2: 3 4	λ 0 1: 1 2 2: 3 4 3: 5	λ 0 1 1: 1 2 6 2: 3 4 3: 5

Figure 10: (a). Heap-tableau P_σ. (b). (Standard) heap-tableau Q_σ.

There are two things to prove about the algorithm outlined above:

(i). For every permutation σ, Q_σ is a heap tableau of the same shape as heap tableau P_σ. Moreover,

Lemma 4. *Q_σ is a heap tableau in standard form.*

(ii). One can uniquely identify permutation σ from the pair (P_σ, Q_σ).

(i). The fact that the shape is the same is easy: whenever number $\sigma(i)$ is inserted into P_σ, this table changes by exactly one (filled) position. When i is inserted into Q_σ, the position on which it is inserted is the unique position that was added to P_σ: the position of the final insertion after a (perhaps empty) sequence of bumps. Therefore the two heap tableaux have the same shape throughout the process, and at the end of it.

Let us show now that Q_σ is a heap tableau. We will show that invariants (b),(c). remain true throughout the insertion process.

They are, indeed, true at the beginning when $Q_\sigma = [1]$. Proving the heap invariant (b). is easy: numbers are inserted into Q_σ in the order $1, 2, \ldots, n$. Each number is, therefore, larger than any number that is an ancestor in its heap. As each number i is inserted as a leaf

in its corresponding heap, all heap conditions are still true after its insertion.

The vector invariant (c). is equally easy: number i is appended to an old vector or starts a new one. The second case is trivial. In the first one i is the largest number inserted so far into Q_σ, therefore the largest in its vector.

Finally, the fact that Q_σ is a standard tableau follows from the Algorithm: Schensted-HEAP$_k$ starts a new vector from the leftmost position available. Therefore when it starts a new vector, its siblings to the left have acquired a smaller number, as they were already created before that point. Also, when it starts a new vector, all the vectors on the level immediately above have been created (otherwise Schensted-HEAP$_k$ would have started a new vector there) and have, thus, acquired a smaller number.

(ii). This is essentially the same proof ideea as that of the Robinson-Schensted correspondence for ordinary Young tableaux: given heap tableaux P, Q with the same shape we will recover the pairs $(n, \sigma(n))$, $(n-1, \sigma(n-1)), \ldots, (1, \sigma(1))$ in this backwards order by reversing the sequences of bumps. We will work in the more general setting when P contains n distinct numbers, not necessarily those from 1 to n. On the other hand, since Q is standard, Q will contain these numbers, each of them exactly once.

The result is easily seen to be true for $n = 1, n = 2$. From now on we will assume that $n \geq 3$ and reason inductively.

Suppose n is in vector V_λ of Q_σ. Then the insertion of $\sigma(n)$ into P_σ did not provoke any bumps. $\sigma(n)$ is the integer in vector V_λ of P_σ sitting in the same position as n does in Q_σ. Suppose, on the other hand, that n is in a different vector of Q_σ. Then n is the outcome of a series of bumps, caused by the insertion of $\sigma(n)$.

Let x be the integer in P_σ sitting at the same position as n in Q_σ. Then x must have been bumped from the parent vector in the heap-table by some y. y is uniquely identified, as the largest element smaller than x in that vector. There must exist a smaller element in that vector by the heap invariant, so y is well-defined. Now y must have been in turn bumped by some z in the parent vector. We identify z going upwards, until we reach vector V_λ, identifying element $\sigma(n)$.

Example 5. *Consider, for example the case of $n = 6$ in Figure 10. Element 2 in P_σ (sitting in the corresponding position) must have been bumped by 1 in the top row. Therefore $\sigma(6) = 1$.*

Now we delete $\sigma(i), i$ from the two heap tableaux and proceed inductively, until we are left with two tables with one element, identifying permutation σ this way.

What allows us to employ the induction hypothesis is the following

Lemma 5. *Removing the largest element n from a standard heap tableau T yields another standard heap tableau.*

Proof. Suppose n is in a vector of length at least two. Clearly, by removing n all the vectors in the heap remain the same, so the resulting table is standard.

Suppose, therefore, that n is the only element in a vector V_β of T, $\beta = zb$, $b \in \Sigma_k$. Since T was standard, all the left sibling vectors V_{za} of V ($a \in \Sigma_k, a < b$) are nonempty, and all the vectors on previous levels of T are nonempty.

Removing V preserves these properties (its leftmost sibling becomes the last vector, or the level disappears completely).

□

□Completing the proof of Lemma 5 also completes the proof of Theorem 8.

□